# Comparison of static- and dynamic call graphs

**Endre Horváth and Judit Jász**

The construction of the call graph is one of the most important tasks during the static analysis of a program. In object-oriented languages, like C++, the presence of pointers and polymorphism can make the results of the analysis imprecise, and the efficiency of the tool or algorithms that use these results can significantly drop.

To make the call graph more precise we have to refine the results from our static analysis. For example we can use some well known algorithms, like CHA or RTA, to resolve the virtual function calls in the program, but in most cases even these algorithms will not give us the most precise call graph. Also we could use a precise pointer analysis, to handle the use of pointers, but in case of large programs, a context- and flow sensitive points-to algorithm requires a large amount of storage space and running time. Generally we can use these accurate pointer analyses only on small systems.

In many cases, tough, it is enough to calculate the call graph from the execution of the program for a set of test cases. This way we do not have to compute the precise call graph, but we can get a set of dynamic call graphs to approximate their static counterpart. We have to notice that the cost of building a dynamic call graph greatly increases with the size of the analyzed program. So we have to find the tools that can build the static and dynamic call graphs of a large system under acceptable circumstances.

We use a program analysis tool for C/C++, called Columbus [1]. It is capable of analyzing real-life, large software products and building the structure of the program in the form of an abstract syntax tree. With the use of Columbus we have different algorithms to compute the static call graph of such programs. We have developed a tool as part of Columbus what can instrument the source code of the analyzed programs, and can make the programs capable of generating their own dynamic call graph. Having run the code, what was instrumented this way, we can get the set of dynamic call graphs and we can use these graphs to approximate the static call graph for the whole system.

We have analyzed some large, real-life software, and have made their static and dynamic call graphs. We have compared these graphs, and have observed whether the static call graphs can be approximately calculated using their dynamic counterparts.

### References

[1] R. Ferenc and Á. Beszédes, M. Tarkiainen, and T. Gyimóthy. Columbus – Reverse Engineering Tool and Schema for C++, *Proceedings of the 18th International Conference on Software Maintenance (ICSM 2002)*, IEEE Computer Society.